Składni C++ ciąg dalszy
Operatory przypisania i de/inkrementacji, funkcje, instrukcje sterujące

Marcin Michalski

ZMP 2024

06.03.2024 WMAT PWr

Przypisanie

- `x = wyrażenie;`
- `x` □`= wyrażenie` oznacza `x = x` □ `wartość`, gdzie □ to dowolny z operatorów dwuargumentowych `+`, `-`, `*`, `/`, `%`, `«`, `»`, `&`, `|`, `^`.

Przypisanie

- `x = wyrażenie;`
- `x` □`= wyrażenie` oznacza `x = x` □ `wartość`, gdzie □ to dowolny z operatorów dwuargumentowych `+`, `-`, `*`, `/`, `%`, `«`, `»`, `&`, `|`, `^`.

Inkrementacja, dekrementacja, pre i post

- Instruckję `x += 1` dla wyrażenia `x` można wykonać za pomocą `++x` i `x++`.
- Różnią się one kolejnością ewaluacji i wykonywania skutków ubocznych.
- Dla `++x` skutki uboczne poprzedzają ewaluację, w przypadku jest `x++` na odwrót.
- Analogicznie dla `--x` i `x--`.

Operatory przypisania, inkrementacji i dekrementacji

Przykład

Jakie wartości pod koniec przyjmą zmienne a,b,c,d w poniższym kodzie?

```
#include <iostream>
using namespace std;
int main () {
  int a=0, b=++a, c=a++, d=++ ++a;
  return 0;
}
```

Dygresja o kolejności

Uważka

Jeśli nie wynika to z konstrukcji instrukcji, kolejność ewaluacji wyrażeń może być nieokreślona.

Przykłady

Demonstracja, kody będą też na stronie.

Funkcje - składnia

Składnia funkcji

```
zwracany_typ nazwa_funkcji(parametry) {
  instrukcje
}
```

- Jeśli `zwracany_typ` to nie `void`, to dla zakończenia działania funkcji i zwrócenia odpowiedniej wartości trzeba użyć słowa kluczowego `return`.
- Nazwy argumentów nazywa się parametrami formalnymi.
- Parametry są opcjonalne.

Przykład

```
void fun() {
  ;
}
```

Przykład

```
int my_add(int x, int y) {
  return x+y;
}
```

Przekazywanie argumentów

Argumenty do funkcji można podać przez

- Wartość – funkcja operuje na kopiach podanych argumentów w nowo przydzielonej pamięci. Pamięć jest zwalniana po zakończeniu działania.
- Referencję – trzeba dopisać za nazwą typu &. Wówczas funkcja operuje dokładnie na argumentach podanych do funkcji, tylko inaczej nazwanych. Nie potrzeba alokować dodatkowej pamięci.

Wartość vs referencja

Demonstracja, kod będzie też na stronie.

Składnia if - else if - else

```
if (wyrażenie) {
  instrukcje_1;
} else if ...
} else {
  instrukcje_n;
}
```

- Jeśli `wyrażenie == true`, to wykonywane są `instrukcje_1`, w przeciwnym razie sprawdzany jest warunek w najbliższym `else if` (jeśli występuje) lub `else`.
- `else if` i `else` są opcjonalne.

Przykład użycia `if ...`

Demonstracja, kod będzie też na stronie.

Składnia switch

```
switch (wyrażenie_całkowito-liczbowe) {
  case const_1:
    instrukcje;
  case const_2:
    instrukcje;
  ...
  default:
    instrukcje;
}
```

switch cd.

- Jeśli `wyrażenie_całkowito-liczbowe` pasuje do case'u, to wykonywana jest odpowiednia instrukcja.
- Może wykonać więcej, niż jeden, case;
- Jeśli żaden case nie zachodzi, to wykonywane są instrukcje występujące po `default`. `deafault` jest opcjonalny.

Przykład użycia `switch`

Demonstracja, kod będzie też na stronie.

Składnia while

```
while (warunek) {
  instrukcje;
}
```

- Jeśli warunek jest prawdziwy, to wykonuje instrukcje, w przeciwnym razie wychodzi z pętli.
- Może nie wykonać ani jednego obrotu.

Składnia do while

```
do {
  instrukcje;
}
while (warunek);
```

- Jeśli warunek jest prawdziwy, to wykonuje `instrukcje`, w przeciwnym razie wychodzi z pętli.
- Wykona instrukcje przynajmniej raz.

Składnia for

```
for (instrukcja_ini; warunek; instrukcja_iter) {
  instrukcje;
}
```

- Wykonuje instrukcję_ini i jak długo warunek jest spełniony, to wykonuje instrukcje.
- Instrukcja_iter jest wykonywana na koniec każdej iteracji.

Instrukcje skoku

- `break` przerywa działanie `switch`'a/pętli i przenosi sterowanie poza blok tych instrukcji.
- `continue` przerywa obecną iterację i przechodzi do następnej. W przypadku pętli `for` dotyczy to instrukcji wewnątrz bloku, `instrukcja_iter` nadal jest wykonywana.

Instrukcje skoku

- `break` przerywa działanie `switch`'a/pętli i przenosi sterowanie poza blok tych instrukcji.
- `continue` przerywa obecną iterację i przechodzi do następnej. W przypadku pętli `for` dotyczy to instrukcji wewnątrz bloku, `instrukcja_iter` nadal jest wykonywana.

Przykłady

Wiadomo.